Tetris AI Using Expectimax Search

He Ma

Abstract

In this project, I applied Expetimax algorithm to an AI, which is call EL Tetris, to the video game Tetris that maximize the total lines eliminated. To compare the performance of the AI using Expetimax Algorithm with the original AI, two experiments were conducted : 1) searching the game states of a reduced height board to an arbitrary depth using top k MAX nodes 2) searching the game states of a standard board to an arbitrary depth using top k MAX nodes depending on the max column height condition. The results show that there is a small improvement to the Tetris AI by applying the Expetimax Algorithm even to limited depth(2 or 3) and extending small number of MAX nodes(2 or 3). I would expect that the performance of Expetimax algorithm will be better if I search deeper and extend more MAX nodes in the Expetimax algorithm.

Tetris is a single player game tile-matching puzzle game that was originally designed and programmed by Alexey Pajitnov in 1984. The standard game board is 10(weight) by 20(height). Each Tetris piece, named Tetromino, takes up 4 unit squares. By convention, people refer to the different shapes of Tetrominoe as Z, S, T, O, J, L, I. One Tetromino will fall from the top of the game board at a time, with equal chance of being any one of the 7 shapes, until it lands on the floor or another Tetromino. While the piece is falling, player can move it to the left or to the right, and turn it 90 degrees, 180 degrees or 270 degrees counterclockwise. This sequence of operations is called one move. When a row is full, it is removed from the board and points are given to the player. If the columns in the game board get too high that the next piece can't fit to the game board, the game terminates.

Tetris can't be played forever. As proposed by Burgiel in his paper How to Lose at Tetris[1], if the S piece and Z piece comes up alternatively, the game will always end before 70,000 tetris pieces are played. It is also proved that the problem of maximizing the number of cleared rows is NP-complete for an offline version of Tetris when the whole Tetromino sequence is known[2]. The goal of my project is to build an AI which maximizes the number of cleared rows for the inline version of Tetris. Only the currently dropping Tetromino piece is known in this case. The goal of my AI is to clear as many lines as possible before the game ends.

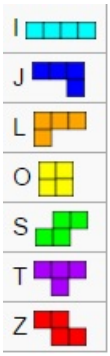

Figure 1: All possible Tetromino shapes[3]

Project Description

My AI is built on top of an AI called EL Tetris[5], which was the best AI for Tetris line elimination until 2014, when Qing Li(Internet ID Misakamm)[4] claimed to built a better one. A game state of Tetris is defined by the status(taken or not) of each square on the game board. So there are 2^{200} game states for a standard Tetris game board. EL Tetris chooses the best move by rating all possible game states after moving the current Tetromino using an evaluation function. The evaluation function is a weighted sum of 6 features: landing height, rows eliminated, row transition, column transition, number of holes, well sum. The weights are determined by applying an optimization method call particle swarm optimization. This evaluation function is used as the search heuristic for my Expetimax search algorithm to depth n, which picks the best move by taking the next n pieces into account.

The Expetimax search algorithm starts with a MAX node holding the current game state and the current Tetromino shape. A MAX node searches for the best position of a Tetromino piece for a given game state. If the MAX node is at depth 0, it returns the best heuristic of the resulting game state after a legal Tetromino move. Otherwise, it returns the best evaluation returned by a EXPECTATION node for each resulting game state after a legal Tetromino

move. A EXPECTATION node assumes that the next piece has a equal chance to be any of the shape, and returns the expected value given by a MAX node on the game state for each of the possible Tetromino shape. Let s be a game state, p be a Tetromino shape, d be the depth of the node, m be a legal move, $rs_{s,m,p}$ be the resulting game state from s by moving p following m. So we have:

$$MAX(s,p,0) = MAX_m eval(rs_{s,m,p})$$

$$MAX(s,p,d) = MAX_m EXP(rs_{s,m,p},d)$$

$$EXP(s,p,d) = \sum_p MAX(s,p,d-1)/|p|$$

One thing to note here is that when we use the Expetimax search algorithm with depth equal to 0, it is exactly the same as EL Tetris. So we can see EL Tetris as a special case of applying the Expetimax search algorithm.

The branching factor for a EXPECTATION node is 7. The branching factor for a MAX node can be 9 to 34 depending on the shape of the piece. So when the search depth is increased by 1, the total number of nodes expended will be increased by 147 on average. In order to search deeper in the tree, a greedy approach is applied to the MAX nodes. Before expending any EXPECTATION child node of a MAX node, the heuristic of the game state for each child is calculated. Only the top k children that has the highest heuristic value will be generated as the EXPECTATION node and expanded. So:

$$MAX(s,p,d)_G = MAX_{top\ m_k(eval(rs_{s,m,p}))} EXP(rs_{s,m,p},d)$$

In this case, the branching factor is $(7k)^d$

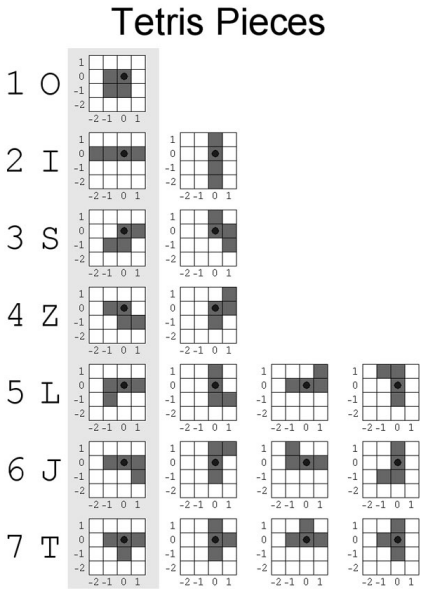

Figure 2: All possible Tetromino orientations[3]

Analysis of Results

I used the emulator implemented by Colin Fahey[4] and improved by Qing Li[3] to test out the performance of EL Tetris and the Expetimax algorithm. Following the design requirements of the emulator, I defined each Tetromino as a 4 by 4 grid as shown in figure 2. A player can only move the piece when the full 4 by 4 grid shows up on the game board. The piece will only locate at the 9th, 10th, 11th and 12th columns when it shows up. A more constraint set is of legal moves is used. A player should rotate the piece first and then move it to the left or right once the piece shows up. When the piece begin to drop, the player can't move the piece anymore. The following two experiments are conducted:

- Lines eliminated from a reduced height board.
  The average number of lines eliminated is tested out on a 10 by 7 game board for Expetimax search of depth 0(EL Tetris), depth 2 with k=2, and depth 3 with k=3. It approximate the scenario that the columns of a standard board gets really high that there is limited space on the game board.

	d = 0	d = 2, k = 2	d = 3, k = 3
Rows eliminated	11	14	15
#experiemnts	14957	924	870

Table 1: Rows eliminated on a 10 by 7 game board

- Lines eliminated from a standard game board.
  EL Tetris can eliminate around 1,000,000 rows in 20 minutes on average when tested on a i7 computer with 8G of memory. The Expectimax search get slower as greater depth and k are used. For a search depth of 3 and k = 3, the average number of lines cleared drops down to 5 lines/second. In order to show the effect of the lookahead with a reasonable amount of test data. I added another parameter $threshold$ in the Expectimax algorithm. For each game state, if the maximum height of the columns in the game board is greater than the $threshold$, call Expectimax with depth n. Otherwise, call Expectimax with depth 0(EL Tetris).

	Rows eliminated	#experiemnts
d = 0(EL Tetris)	896,587	569
d=2,k=2,t=16	1,103,177	284
d=2,k=2,t=15	1,093,452	244
d=2, k =2, t = 14	1,040,917	199
d=2, k =2, t = 13	972,113	189
d=2, k =2, t = 12	1,032,544	135
d=2, k =2, t = 11	1,115,044	108
d=2, k =2, t = 10	1,329,432	122
d=2, k =2, t = 9	1,294,576	99
d=2, k =2, t = 8	1,134,586	73
d=2, k =2, t = 7	1,289,405	57

Table 2: Rows eliminated on a standard game board

Below is a plot that shows the percentage of time searching to depth 0 and searching to depth 2 yields different result

when running Expectimax with d=2, k=2,t = 10. 1,141,341 lines were cleared. 13,417 game states have a max height that is greater or equal to 10.

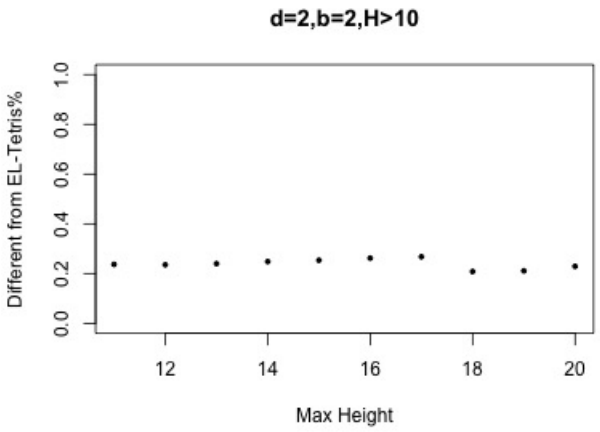

Figure 3: The effect of searching deep. Searching to depth 2 yields a different result from search to depth 0 20% of times.

Discussion

Figure 3 suggests that searching deeper has an effect on the decision made by the AI. From the data in Table 1 and Table 2, we can see that the average number of rows cleared is always higher than EL Tetris when search depth is greater than 0. By conducting a hypothesis testing, I can prove that both depth = 2 and depth = 3 models are different from the EL Tetris model on the 10 by 7 game board. However, the amount of improvement is not as significant as I expected for a standard game board. I noticed that the variances of the total number of lines eliminated for the models are big when tested on the standard board. Figure 4 is a plot for the total number of lines eliminated by model d=2, k=2, t=10. The standard deviation is almost the same as the average of the dataset. And the value of an outlier could have a huge impact on the average. So I'm not confident yet to draw the conclusion that Expetimax algorithm performs better than the original EL Tetris. At least 10 times the amount of data I have now is needed in order to prove that the Expetimax model outperform EL Tetris on a standard game board.

Conclusion

The performance of Expetimax algorithm is better than EL Tetris when the game board is 10 by 7. For a standard game board, more data is needed before drawing the conclusion that Expetimax algorithm is better than EL Tetris.

In the future, I would like to optimize the Expectimax algorithm by caching the search result, so that I could test it with greater depth and k. Also I would like to test the performance of Expectimax algorithm for Tetris games with the next few pieces revealed.

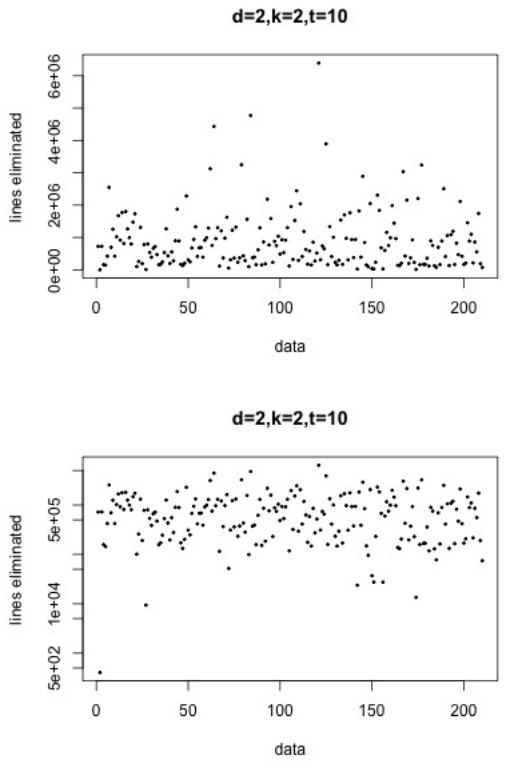

Figure 4: total number of lines eliminated by model d=2, k=2, t=10; Bottom: same plot as the top one with y axis in log scale.

In addition, I learned that problems using Expectimax algorithm can also be solved using Markov Chains. There might be a way to reduce Tetris to a problem with fewer game states, so that we can study the problem using Markov Chains.

Reference

1. Heidi Burgiel. 1997. How to Lose at Tetris.
2. Erik D. Demaine, Susan Hohenberger, David Liben-Nowell. 2002. Tetris is Hard, Even to Approximate.
3. Colin Fahey. 2003. http://www.colinfahey.com/tetris/tetris_en.html.
4. Qing, Li. 2014. https://misakamm.com/blog/504.
5. Islam El-Ashi. 2011. http://ielashi.com/el-tetris-an-improvement-on-pierre-dellacheries-a